UART Data Transmit with STM32F0

Remember when printers, mice, and modems had thick cables with those huge clunky connectors? The ones that had to be screwed into your computer? Those devices were probably using UARTs to communicate with your computer. While USB has almost completely replaced those old cables and connectors, UARTs are not a thing of the past. You'll find UARTs being used in many DIY electronics projects to connect GPS modules, Bluetooth modules, and RFID card reader modules to your Raspberry Pi, STM32, or other microcontrollers.

UART stands for Universal Asynchronous Receiver/Transmitter. It's not a communication protocol like SPI and I2C, but a physical circuit in a microcontroller, or a stand-alone IC. A UART's main purpose is to transmit and receive serial data. One of the best things about UART is that it only uses two wires to transmit data between devices.

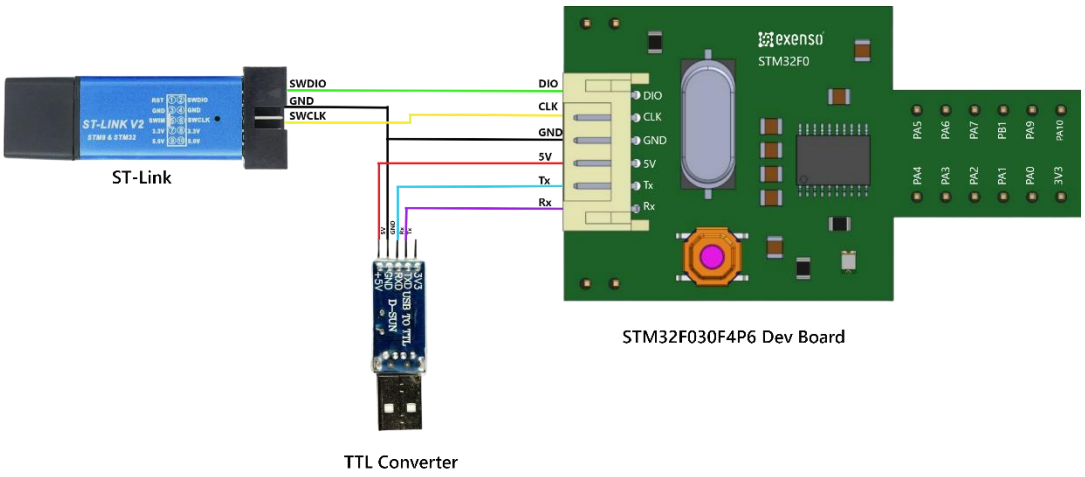
Components Required

You will need the following components –

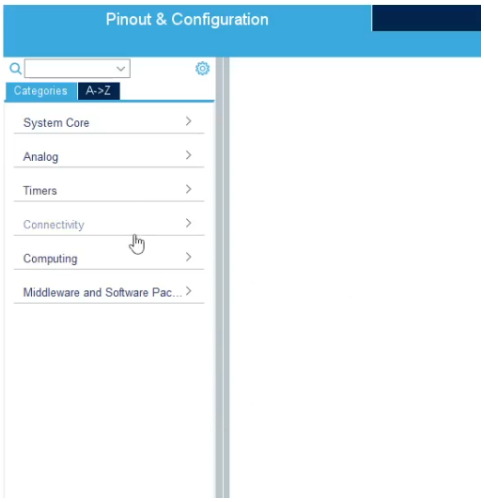
- 1 × Breadboard
- 1 × STM32F030F4P6
- 1× TTL Converter

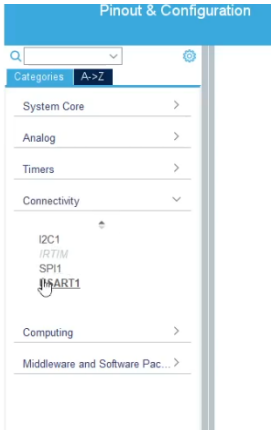
Procedure

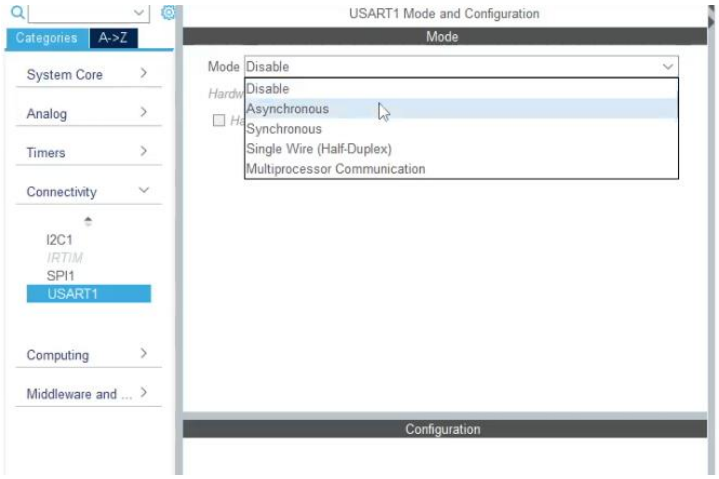
Follow the circuit diagram shown in the image given below.



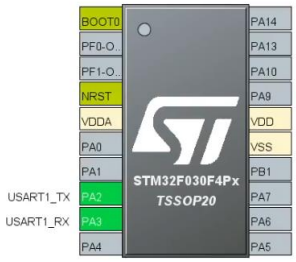
STM32F0 Pin Configuration:



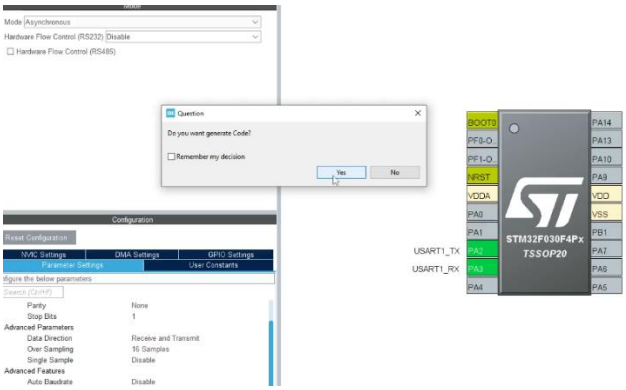





Mode Asynchronous
Hardware Flow Control (RS232) Disable
Hardware Flow Control (RS485)
Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Configure the below parameters :
Search (Ctrl+F)
Basic Parameters
Baud Rate 38400
Word Length 8 Bits (including Parity)
Parity None
Stop Bits 1
Advanced Parameters
Data Direction Receive and Transmit


BOOT0
PF0-O..
PF1-O..
NRST
VDDA
PA0
PA1
PA2
PA3
PA4
PA14
PA13
PA10
PA9
VDD
VSS
PB1
PA7
PA6
PA5
USART1_TX
USART1_RX
STM32F030F4Px
TSSOP20


Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Configure the below parameters :
Search (Ctrl+F)
Basic Parameters
Baud Rate 9600 Bits/s
Word Length 8 Bits (including Parity)
Parity None
Stop Bits 1

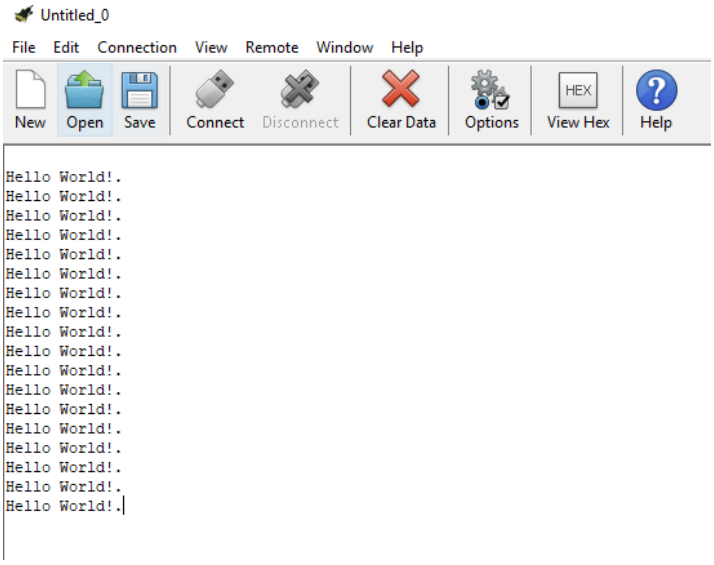

Mode Asynchronous
Hardware Flow Control (RS232) Disable
Hardware Flow Control (RS485)
Question
Do you want generate Code?
Remember my decision
Yes
No
Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
Parity None
Stop Bits 1
Advanced Parameters
Data Direction Receive and Transmit
Over Sampling 16 Samples
Single Sample Disable
Advanced Features
Auto Baudrate Disable
USART1_TX
USART1_RX
STM32F030F4Px
TSSOP20

Code

Uart String data Transmit code

```
#include "main.h"

UART_HandleTypeDef huart1;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);

uint8_t str_buff[]="\nHello World!";

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART1_UART_Init();
  while (1)
  {
        HAL_UART_Transmit(&huart1, str_buff, sizeof(str_buff), 1000);
         HAL_Delay(1000);
  }
}
```

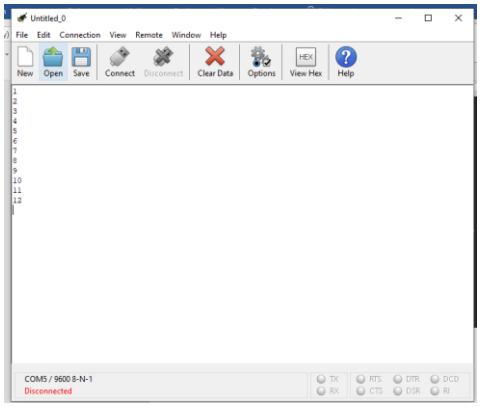
Output



Uart integer data Transmit code

```c
#include "main.h"

UART_HandleTypeDef huart1;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);


void Int_to_str(int digit, char string[]) {
  int i = 0;
  while (digit > 0) {
    string[i++] = (digit % 10) + '0';  // Convert digit to character and store
    digit /= 10;                       // Divide by 10 for the next digit
  }

  // Reverse the string to get the correct order
  for (int j = 0; j < i / 2; j++) {
    char temp = string[j];
    string[j] = string[i - j - 1];
    string[i - j - 1] = temp;
  }
  string[i] = '\0';  // Add null terminator
}



int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART1_UART_Init();

  uint8_t Ivalue = 0;       // Init int value
  char int_buff[4];         // Store int value
  char next_line[2] ="\n";  // next line

  while (1)
  {

      Ivalue ++;
      // conversion int to a string value
      Int_to_str(Ivalue,int_buff);
        // transmit string value
      HAL_UART_Transmit(&huart1, int_buff, strlen(int_buff), 1000);
      // for next line
      HAL_UART_Transmit(&huart1, next_line, strlen(next_line), 1000);
      // Delay 1000ms or 1s
       HAL_Delay(1000);


  }

}
```

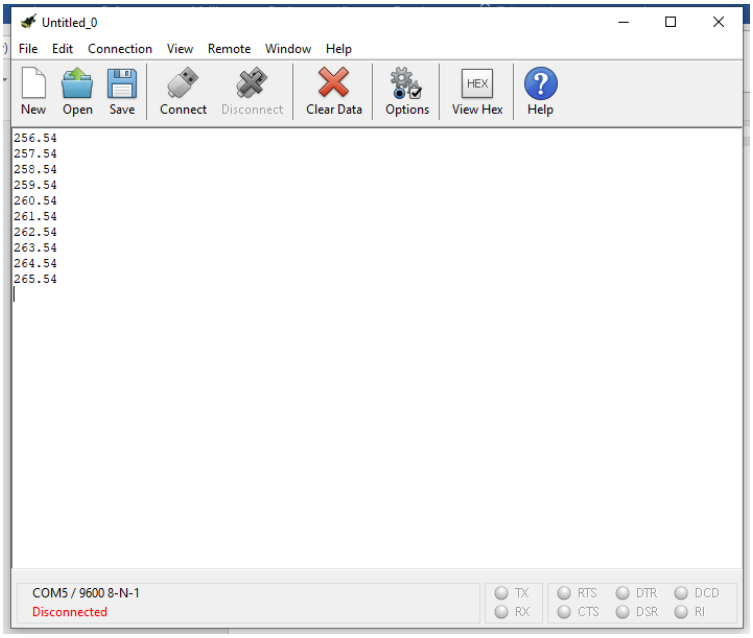
Output



Add “Print_Float.h” header file for Uart Float Data Transmit

```c
#include "main.h"
#include "Print_Float.h"

UART_HandleTypeDef huart1;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART1_UART_Init();
  float Fvalue = 245.54821;      // Init float value
  uint8_t float_buff[20];      // Store int value
  uint8_t next_line[2] = "\n"; // next line
  while (1)
  {
        Fvalue++;
        // float to String Conversion
        ftoa(Fvalue, float_buff, 2);
        // transmit string value
        HAL_UART_Transmit(&huart1, float_buff ,strlen(float_buff), 1000);
        // for next line
        HAL_UART_Transmit(&huart1, next_line,strlen(next_line), 1000);
        // Delay 1000ms or 1s
        HAL_Delay(1000);

  }

}
```

Output




```
HAL_UART_Transmit(UART_HandleTypeDef *huart, const uint8_t *pData, uint16_t Size, uint32_t Timeout)
```